Safeguarding and Charging for Information on the Internet[*]

Hector Garcia-Molina, Steven P. Ketchpel, Narayanan Shivakumar
Stanford University
Computer Science Department
Stanford, CA 94305
{hector, ketchpel, shiva}@cs.stanford.edu

Abstract

With the growing acceptance of the Internet as a new dissemination medium, several new and interesting challenges arise in building a digital commerce infrastructure. In this article we discuss some of the issues that arise in building such an infrastructure. In particular, we study how one can find and pay for digital information, and how one can safeguard the information from invalid access and duplication. We use examples from our Stanford Digital Library Project to illustrate some of these problems and their potential solutions.

1 Introduction

As the the Internet population has expanded beyond the scientific research community, businesses have recognized the potential of electronic commerce. The Internet makes large number of customers available, and electronic commerce could reduce the costs of business-to-business sales by eliminating much of the paperwork and bureaucracy. On the other hand, customers also like the appeal of electronic commerce, with many more products available and the possibility of easily shopping for the lowest prices.

However, the infrastructure for e-commerce has lagged behind the interest of buyers and sellers. On the one hand, there needs to be widespread availability of client software for the individual customers to find, purchase, and receive the desired goods over the network. On the other hand, server software for merchants needs to automate both "front office" activities of presenting the goods and taking orders, and the "back office" activities of processing payments, and integrating with accounting, inventory and order fulfillment systems. Furthermore, when it comes to electronic goods (documents, music, movies), safeguarding mechanisms are needed, making it hard for users to access the goods without paying or for them to make illegal copies.

In this paper we discuss some of the issues involved and potential solutions for electronic commerce, focusing on the infrastructure for finding, paying, delivering, and safeguarding electronic goods or information. In addition, we highlight some of the work in this area that has been carried out as part of the Stanford Digital Library Project.

We start by describing the main components of an electronic commerce system, i.e., ordering, payment and delivery systems. Then in Section 3 we discuss how these components can be combined into complete solutions. Within each solution, some of the significant providers and tools will be presented. In Sections 4 and 5 we describe Stanford research projects in this area, mainly the Universal Payment Application Interface (U-PAI) and the Shopping Model Architecture. We focus in more detail on the content delivery problem in Section 6, discussing how information can be protected, how illegal copies can be detected, and how illegal copies can be traced back to the original copy they were made from. In Section 7 we then overview the Stanford Copyright Analysis Mechanism (SCAM), which provides efficient copy detection.

2 E-Commerce Components

Customers using e-commerce applications typically begin their shopping transaction by searching for a product that they want, or browsing through an electronic storefront or catalog. They may view descriptions or pictures of goods, ultimately selecting some for purchase. Following their selection, they must be able to pay for their goods, and, in the event that the goods are digital, receive and use the goods. These steps of placing an order, paying for the order, and receiving a delivery correspond to the main services required of e-commerce client software. The merchant needs to have server software that supports the dissemination of product information and collection of orders; the processing of customer payments; and the delivery of goods. These processes will be described in turn in the following subsections.

2.1 Ordering Systems

Given the diversity of end-user systems in widespread use, creating a special purpose application to connect a user to an electronic storefront is beyond

[*]

the resources of most merchants. Fortunately, it is not necessary. The existing infrastructure of web browsers provides a reasonably sophisticated platform for the presentation of goods. General purpose browsers permit the display of textual and graphic information and simple animation in a consistent format to almost all users regardless of their personal system. More engaging information, such as audio, full video and "applets" may be presented to consumers at the higher end of the technical spectrum.

Since each merchant has its own electronic storefront and process for selecting goods and placing orders, there had been little standardization of this process. As long as the majority of transactions are between an end-user and a single merchant's system, the gains of standardization and interoperation are limited. In the future though, as "intelligent agents" act on behalf of shoppers, a programmatic interface to the ordering system will provide greater value. The BargainFinder agent [18] required hand-coded modifications to interact with each new CD store to find the lowest price for a piece of music. Jango [21], by Netbot, Inc., partially automates the process of creating "adaptors" to interact with different stores.

2.2 Payment Systems

The financial infrastructure for the open network environment was a largely academic concern until the early 1990's. A proposal for digital cash [4] existed in the 1980's, but was not implemented for ordinary use. The widening audience of the Internet and specifically the World Wide Web showed a great deal of commercial promise, and the lack of a payment mechanism became an obvious business opportunity. The rush to create a practical system was on, and the mid-1990's saw the launch of systems such as First Virtual [38], CyberCash [5], e-cash from DigiCash [7] and the Secure Electronic Transaction (SET) protocol from Mastercard and Visa [29]. These systems shared the goal of moving money between accounts, but varied widely in their methods. Some are credit-based, others debit-based, some rely on cryptography to provide security, others forgo it. They use a diversity of communication protocols ranging from secure versions of HTTP to e-mail. Some provide a "wallet" user interface, others do not. Comparisons between the systems have been published elsewhere [27]. The interested reader is referred to the original papers for the technical details of the protocols and to these comparison papers for the feature sets and other considerations to be used in choosing among them.

Given all these schemes, the merchant had the option of selecting among several payment systems providers. The reality, however, is that none of the systems had a significant user base, and transactions were small in number and value. Sending credit card information in plain text, across an encrypted socket, or out-of-band were the preferred methods for customers to pay for goods they ordered on line.

2.3 Delivery Systems

For many information goods, the network became not only a convenient way to find goods and place and pay for an order, but also a fast, efficient way to deliver the goods to the purchaser. However, retailers were concerned about their intellectual property assets, recognizing that a perfect digital copy could be made by an eavesdropper or buyer and re-distributed easily without the appropriate fees being paid to the author or publisher of the information. Section 6 describes the issues involved in content delivery.

3 Combining Components

The components described in the previous sections are largely stand alone services. A merchant can receive a payment with First Virtual, or send a document with a secure delivery service. Tying these two activities together, or presenting the goods for immediate purchase are beyond the scope of any of these components individually.

In order to fill this void, a number of companies, both new and established, are offering solutions to the merchant wishing to enter the e-commerce world. The solutions fall into three types: "buy", "build", or "rent". First, merchants can buy and operate a complete commerce server. Second, they can build their own system by combining the components themselves. Third, they can employ a service provider to manage the e-commerce system, essentially outsourcing the data processing requirements while focusing on the content generation. These three solution styles are described in the following subsections.

3.1 Commerce servers: The "Buy" Approach

Web servers were designed to facilitate the sharing of information, using relatively simple protocols that would be easy to implement on a number of platforms. These systems were not intended for e-commerce however. The goal of widespread dissemination of scientific information was incompatible with the practice of charging for information or goods.

In order to accept payments for subscriptions or individual content items, the main developers of web servers extended their products into the arena of "commerce servers", with the Netscape Enterprise Server [22] and Microsoft's Merchant Server [19]. Oracle saw an opportunity to leverage their database business into this area as well [24]. Startup companies such as iCat [14] focused on the niche, exhibiting strength in areas such as producing a catalog from a database. NetBill [36], a research project at Carnegie Mellon University, links the payment with the delivery of electronic goods, providing guarantees about a fair exchange.

These products provide merchants with a complete solution, though of limited flexibility. Typically, the servers accept payment via encrypted credit cards and deliver goods via HTTP (or provide an entry into the merchant's existing fulfillment process for physical goods). Support for other payment protocols and delivery mechanisms is limited, frequently requiring significant development effort on behalf of the merchant.

3.2 Universal Protocol: The "Build" Approach

Recognizing the need to integrate these diverse components into more complete services, a number of industry consortia have proposed the creation of universal interfaces or protocols so that a developer could call upon any one of a number of external services without having to tailor the application to a novel interface each time. Similarly, by providing support for the universal interface, other developers could make use of the application more easily. Examples in this category include Java's JECF [39], SEMPER [28], eCo [40], and the Open Trading Protocol [20].

3.3 Outsourcing: The "Rent" Approach

For smaller merchants especially, the prospect of running a commerce server or developing their own e-commerce applications is daunting. They wish to reach customers in cyberspace without building or buying all of the support systems for handling electronic commerce. To serve these firms, a number of providers have offered to outsource e-commerce services. For example, AT&T offers SecureBuy [2], (using the Open Market Transact system [23] to provide the back office functions), where merchants can permit AT&T to handle all of the system installation and administration, while they fulfill the orders.

4 Payment Mechanism Independence

As discussed in Section 3.2, universal interfaces to e-commerce components are highly desirable. In this section we briefly describe one protocol developed at Stanford, that tries to isolate payment clients from the diversity and complexity of payment mechanisms. The U-PAI (Universal Payment Application Interface) system (and SEMPER's Generic Payment System [1]) provides a standardized payment interface to payment mechanisms. A layer of proxies, which can be implemented by the payment mechanism provider or a third party, translate between the mechanism's native interface and that specified by U-PAI. Therefore, neither the customer's nor the merchant's application code needs to be changed when a new payment mechanism is introduced. The calls to the U-PAI interface that enabled interaction with the old payment mechanisms work for the new one as well.

Defining an appropriate interface requires including the appropriate amount of functionality. If important functions are excluded, application developers will be forced to go outside the standard, and the benefits of the standard are largely nullified. If too much functionality is included, the implementation of the standard becomes difficult or impossible, decreasing its chances of adoption.

An important feature of the U-PAI system is that all of its remote calls are asynchronous and non-blocking, with a callback used to return the result of the remote computation. This feature adds robustness if the remote service is unavailable, and also permits a user to attempt to abort a payment that is in progress.

Figure 1 shows the steps involved in using the U-PAI protocol in conjunction with the e-cash system from DigiCash. Steps A through D are defined by the U-PAI system, and are independent of the details of the e-cash protocol. The steps numbered 0 through 6 are those particular to e-cash. The customer application creates a *Payment Control Record (PCR)* complete with information about the source and destination account, and amount of the transfer. When the `startTransfer` method is invoked and repeated to the *accountHandle* (proxy representing the payment mechanism), the native protocol (e-cash, in this case) is initiated. That protocol proceeds until complete (or a significant checkpoint is reached) when the disposition of the transaction is sent to the `accountHandle` which sends it back to the `PCR` and from there to any *Monitor* that needs to be apprised of the transaction's progress. The merchant may be one such monitor, delivering the goods if the payment was successful. Note that the customer and merchant's applications (and monitors) are completely insulated from the details of the payment mechanism, and could use any other payment mechanism with the appropriate U-PAI interface (`accountHandle` proxies). For more detail on the U-PAI system, consult [17].

5 Shopping Model Architecture

Payment mechanism independence is a good first step to an interoperable system. However, as described above, payment is only one component of an e-commerce application. While equivalent techniques could be used to permit interoperation for delivery systems or ordering systems, a whole different plane of flexibility is achieved by controlling the sequencing among these operations. For example, an annual subscription to a weekly magazine may be implemented as an order and a payment, followed by 52 deliveries. Some merchants may permit goods to be delivered before receiving payment, while others insist on payment first. Sessions can be represented as a number of orders followed by deliveries, with an aggregated payment at the end of the session. These are examples of *shopping models* [16]. The goal of the Shopping Model Architecture is to permit a customer and merchant to select a shopping model that they agree upon, perhaps one that was not even proposed at the time their applications were developed, and carry out the transaction according to that shopping model.

A *Shopping Controller* is an object which implements a particular shopping model, that is, a sequencing among the ordering, payment, and delivery actions. Instead of directly communicating with each other, the merchant and customer send their messages to the `ShoppingController`, which determines the correct "next step" for the transaction, and makes the appropriate call upon the relevant participant. The `ShoppingController` also acts as a `Monitor` (in the U-PAI sense) for the payment and delivery actions between the merchant and customer, and is able to advance the transaction based on the outcome of them.

For example, in Figure 2, a fragment of a transaction is shown, beginning with the merchant's receipt of a complete order. The merchant acknowledges the order (Message 1) by contacting the `ShoppingController`. This controller happens to implement a "Pay First, Deliver Later" model, so

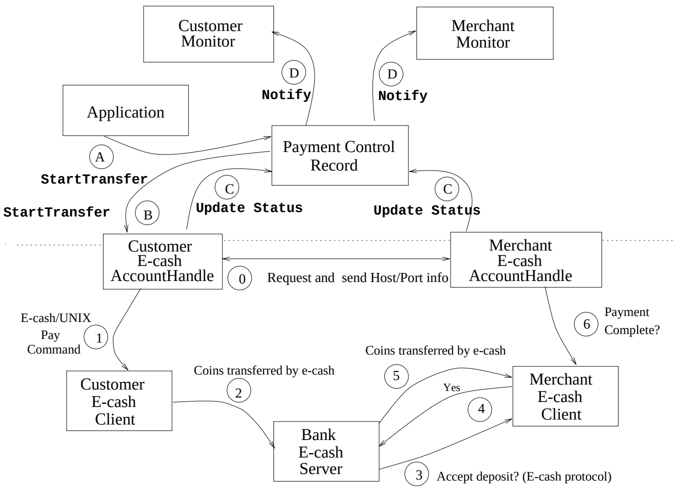

Figure 1: An E-cash Transaction using U-PAI

it instructs the customer to pay by sending a bill (Message 2). The customer invokes U-PAI to transfer money to the merchant's account (Message 3). The `ShoppingController` is a `Monitor` to the U-PAI transfer, so when the transfer completes successfully, the controller is notified (Message 4). In the completion of the example, the controller would contact the merchant, which would send the goods via a delivery protocol similar to U-PAI.

If the `ShoppingController` were a "Deliver First, Pay Later" model, the first message indicating that the merchant had received a complete order would result in the controller invoking a method on the merchant to trigger delivery. The customer and merchant applications simply respond to messages generated by the controller, and are implemented without knowledge of the controller details. Therefore, they are able to select controllers that implement new models and operate under them without modification to the application code. Additional details about the Shopping Model Architecture may be found in [16].

6 Content Delivery

Once a customer has bought some goods and paid for them, the merchant has to deliver the content – in case of digital content, the merchant may use some protocol such as HTTP or e-mail to deliver the goods. However when the provider uses digital means of transmission, he runs the risk of the content being copied and made available to the entire world through the World Wide Web. In such a case, the provider will be paid only once, i.e., for the first copy, while most people will access the content from the alternate web/ ftp source. Hence many book publishers, photographers and movie makers are facing a dilemma regarding the digital availability of valuable content. On the one hand, they would like to use the web for (1) tapping into the "impulsive buyer" market who relish immediate access to required information, and for (2) making higher profits due to lower delivery costs. On the other hand, they will lose revenues due to illegal copies. As a compromise solution to this quandary, book publishers are beginning to offer a few pages of popular novels as a "teaser" on the web, while selling physical copies of the books to the reader.

A related problem arises due to the growing popularity of the web as a dissemination medium. Until recently making an illegal copy of a best-selling book and distributing it widely was very hard for two reasons: (1) the physical act of photo-copying or printing books was time-consuming, and (2) the distribution channels available to the law-breakers were limited. However today any small-time operator with good scanning and OCR technology can reproduce books and make them available on the web, usenet newsgroups or public mailing lists at little cost. Professional photographers and art collectors also face similar problems when images in their portfolios are made freely available to the world.

Protecting digital documents from illegal copying has received a lot of attention recently, both in terms of revised intellectual property laws, as well as new technology-based solutions [26, 11]. For an excellent discussion of issues involved in intellectual property, from a legal perspective refer to [10]. We now outline some of the technology-based solutions currently being developed as research prototypes or available as commercial products in the following subsections.

6.1 Copy Protection

In the copy protection approach, content providers use a variety of techniques to make it very hard to make copies. One common approach is to physically isolate information by placing the information on stand-alone CD-ROM systems, and by letting users access data through some search interface. Another approach is to use special-purpose hardware for authorization [25], or to build "fully trusted systems"

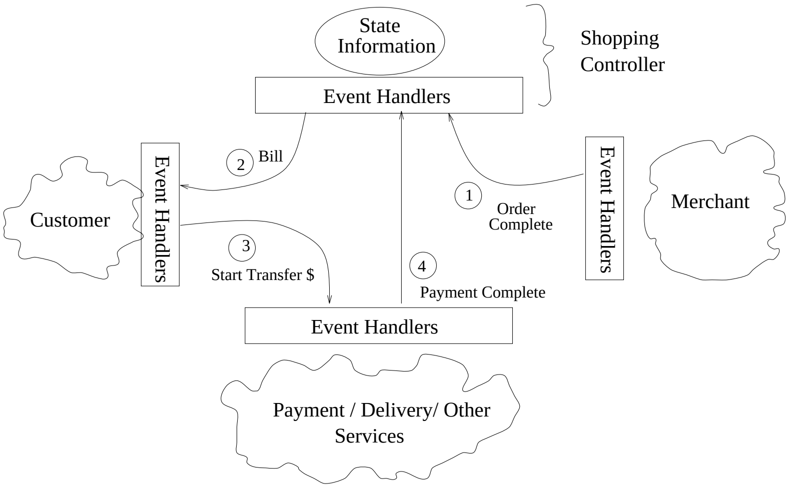

Figure 2: A high level view of a shopping model

using "fully trusted components" and secure interconnections [37].

The software industry, when faced with software piracy, adopted a very rigorous policy of disallowing making disk copies of software. However this policy led to a decline in sales since end users could no longer make backup copies of software to recover from crashes – the software companies withdrew this policy in the mid-1980's. Since then the software industry has tried a variety of schemes to reduce software piracy, but these measures always end up falling short of hackers.

Recently IBM introduced *Cryptolopes* [15]: Cryptolopes are essentially objects containing a Java applet to interact with users, business rules for content usage, and an encrypted version of the content. When users want to access the content, they first pay for the content using the program and obtain a decryption key from a cryptolope clearing center located remotely. They then use a "trusted viewer" to view the content. Intertrust offers a similar product in *DigiBoxes* [35]. The approach adopted by Cryptolopes and DigiBoxes however has the following drawbacks: (1) it requires a user's favorite viewer (such as emacs, vi, Microsoft Word) to be "cryptolope-compatible," (2) hackers can still write software emulators or screen capturing software to obtain a copy of the data once they have paid for the content, (3) it does not solve the problem of a person scanning an image from a physical copy of the magazine (or OCRing its textual content), and offering it on a web site.

6.2 Watermarks

In the watermarking approach, the content provider uses steganographic techniques to hide additional information into digital content. For instance when an image is sold to a customer, the image could be imperceptibly marked with the user's credit card number. Once the illegal version of some content is found on a public web site (using techniques presented in the next subsection) the content provider can then track the original buyer by extracting the watermarks.

Currently there is significant research in adding imperceptible, "hard-to-remove" watermarks to digital content, especially for images and audio [13]. There are also commercial products from Dice [6] to watermark audio, as well as from Digimarc [8] to watermark and track images on the web. However, current techniques are not completely resilient to hackers – in fact hackers may view the problem of breaking the watermarking scheme as a challenge to their hacking skills. Also the watermarking approach does not address the problem of a person scanning an image from a physical copy of the magazine, and making it publicly available.

6.3 Copy Detection

In the copy detection approach, the content provider imposes few restrictions on how content is distributed. However, the content provider finds *illegal* exact copies and partial copies (e.g., chapters or sections of books) of text and images on the World Wide Web (i.e., web sites, newsgroups, mailing lists). by keeping track of who he sells objects to, and who possesses illegal copies. An important and challenging problem in this context is how to automatically find illegal copies of content such as text, images and videos especially as the number of web sites, newsgroups and mailing lists increases. For this, we can build a Copy Detection System (CDS) using the following components:

1. *Crawler:* The crawler crawls through the World Wide Web and presents a stream of documents and images to the CDS.

2. *Registration Repository:* This is a database of objects such as books and images *registered* by book publishers and photographers who want to find illegal copies of their works.

3. *Similarity Matcher:* When the crawler presents a *query* object, we need to efficiently compare the object to objects in the repository. If the query object is a "close-enough" match to some object in the repository, we notify the owner of the registered object of the potential illegal copy.

For example, if Walt Disney Corporation wants to find all images of Mickey Mouse available on the World Wide Web, they would register images of Mickey Mouse into the registration repository of the CDS. When the similarity matcher gets an image from the crawler, it finds all images in the repository that are "close enough" to the input image. If the query image is an image of Mickey Mouse, the similarity matcher should find the corresponding registered images and notify Disney Corporation.

There are many challenges in building an *ideal* CDS. The ideal CDS should be:

1. *Accurate:* Given a query object, the similarity matcher should return the set of objects that are close to the query object according to some similarity measure. That is, the good CDS will have high *precision* and *recall* (using standard terminology from the Information Retrieval community).

2. *Scalable:* The repository and similarity matcher should scale as the number of registered and crawled objects increases.

3. *Resilient:* The similarity matcher should be resilient to any tampering by an adversary who can "guess" how the CDS detects copies. For instance, an illegal copy of a text document modified by adding some words or by changing the format (e.g., Word versus HTML or Postscript) should still be detected as a copy. An illegal copy of an image modified by operations such as cropping, dithering, blurring should still be detected as illegal.

Building such an ideal CDS is technologically difficult in the near future. Today, we can build an efficient CDS that may be inaccurate and not resilient to tampering. For example, we can compute checksums for every registered object and store these checksums into a database. For each query object, we can then compute the corresponding checksum and find objects with the same checksum efficiently. However this scheme cannot handle partial matches and may be broken by the insertion of even a single word into an illegal document, or by cropping an image. Alternately we can build a CDS which performs tests with high precision, recall and resiliency but is not scalable. For example to find illegal partial textual overlap, we can compare every registered document against all known documents, looking at the common maximal subsequences of characters. Documents that share many long subsequences would be considered potential copies. While this may be more accurate and resilient to several forms of tampering (such as inserting words into the text randomly), computing maximal subsequences is expensive, and the scheme does not scale well as the number of query documents and registered documents increases. In Section 7 we present a copy detection scheme that is more efficient, although perhaps less accurate and resilient.

Observe that copy detection systems are also an important component of the watermarking schemes: we need to first find illegal partial and exact copies on the web using a CDS, before watermarks can be extracted to identify the original buyer. In some cases the role of the CDS may be simplified, especially if all partial copies of the content still retain the watermarking information.

7 Overview of Stanford Copy Analysis Mechanism (SCAM)

We have implemented an experimental prototype, the Stanford Copy Analysis Mechanism (SCAM), as a "proof-of-concept" CDS system for text documents. This system indexes Netnews articles and web documents and allows users to check for potential illegal copies of their documents. We have developed a web interface to this system and have also integrated it into the Stanford Digital Library testbed. In the rest of this section we summarize some of the research challenges that were addressed as SCAM was implemented.

The reader may be interested to know that we have used the SCAM system to find actual occurrences of academic plagiarism. In a renowned case, a graduate student had copied and partly modified technical reports of researchers in computer science and resubmitted them under his own name to different journals and conferences. When ACM was compiling a case against him, they had access only to his abstracts. But in order to compile the case, ACM needed to find the set of original technical reports he had copied from. We used SCAM to track these initial technical reports using abstracts from INSPEC, a repository of abstracts of technical papers in computer science. The success of SCAM was reported in several mailing lists and newsgroups on the Internet, as well as magazines such as Communications of the ACM, Forbes Magazine, The Recorder and the Intellectual Property Magazine.

7.1 Chunking

We believe one way to achieve accuracy, scalability as well as resiliency is to *chunk* an object into smaller primitives and to compute similarity measures using these chunks. For example, we can chunk registered text documents into sentences and store these sentences in a database. When a query document is to be checked against the registered objects, we can chunk the query document into sentences and check the database for documents with the same sentences. We can then define a registered document to be a "close enough" copy if it shares a *sufficient* number of sentences with the query document. This form of chunking is accurate for finding documents that have sentence overlap, and is resilient to tampering unless a majority of sentences in the illegal document are modified. Such a CDS is also scalable if we build efficient data structures (such as inverted indices on sentences) to find registered documents containing a given sentence.

In [30, 31] we propose a variety of chunkings (such as words, subsequences, sentences and hashed breakpoints [3]), similarity measures and data structures to compute similarity between textual documents. We also report a comprehensive evaluation of these chunkings with respect to important performance measures such as space consumption, load time, probe time, and false positive and false negative errors on a large document set.

Often one needs to detect copies between a large set of registered documents, and a large set of potential copies. This can be done in "batch" mode, as opposed to comparing each registered document against the potential copies, one at a time. This is analogous to a database join, where every document in one set has to be checked for potential overlap against all documents in the second set. In [32] we present algorithms for efficiently comparing large batches of documents.

7.2 Filtering Expensive Tests

In general, there are many ways to check for overlap between documents. Some strategies may be accurate but expensive (e.g., searching for maximal subsequences), while others may be more efficient but less accurate. Actually, tests could be less accurate in two ways: they can fail to detect documents that did overlap significantly (false negatives), or they can indicate that two documents overlap when in reality they do not (false positives). These tests can be *composed* to obtain tests that have better accuracy and performance characteristics. For example, say we have a test T_1 that is inexpensive but yields many false positives, and a test T_2 that is more expensive but has many fewer false positives. Then, to test a document, we can first run T_1. If the test is positive, then we can run T_2 to confirm the result.

In [34] we propose a general framework for expressing tests in terms of their expenses, errors and selectivity. We show how to compose tests using different operators, and present provably good algorithms for choosing the right set of tests to use, in order to reduce costs and errors. This framework is applicable beyond SCAM, and we have considered how it can be applied in other "fuzzy" matching applications such as in IBM's Query By Image Content (QBIC) image searching engine.

7.3 Selecting Text Databases and Extracting Documents

So far we have assumed that SCAM collects "all" documents for comparison against registered documents. This is fine if the potential copies are freely available, but is not feasible if the potential copies are "hidden" behind search engines, or if we cannot afford to collect all potential copies. For instance, say we are given a document d that may have a copy at one of several databases $B_1, ..., B_n$. Each B_i provides a traditional text search interface, but is incapable of running a copy detection algorithm for us.

A first challenge is to discover which of the B_i databases are good candidates for finer testing. One possible scheme is to generate a *detection* query based on important words in d, or perhaps using all words in d. We can then submit the detection query to each database; if the number of documents in the answer is high we can consider that database a good candidate for further exploration. Another possibility is to rely on statistics about each B_i (e.g., how frequently each words appears) to discover good databases. This approach is similar to that used by GlOSS [12].

Once we have identified B_i as a candidate, a second challenge is to extract from it documents that may be copies. For this, we can generate an *extraction* query, submit it to B_i, and then use SCAM to compare the returned documents with d.

In [9] we have studied these problems in detail. We proposed schemes for detecting good candidate databases based on statistical information, as well as strategies for constructing good extraction queries. Our experimental results show that these combined ideas can be very useful for copy detection when SCAM cannot analyze all documents.

7.4 Indexing Data Windows

In many cases, authors and photographers are interested in checking if their work has been copied in a certain past time period, for example, the past week. This and several other applications in databases motivate the construction of indices that can efficiently index data of a past window of days. For example, if we wish to maintain an index on data of the past 7 days, an interesting question is how to efficiently add a new day's (8^{th} day) data and quickly expire data of the 1^{st} day. An obvious way is to maintain a single index and delete entries of the 1^{st} day, and add entries of the 8^{th} day to the index. However in [33] we propose a family of *wave indices*, which are alternate ways of organizing indices that are optimized for maintaining windows of data. We also show using examples from SCAM, Altavista and a Data Warehousing benchmark how different wave indices may make indexing windows of data more efficient.

8 Conclusion

In this paper we have discussed some of the challenges involved in building a good electronic commerce infrastructure. The problems range from interoperability (e.g., among different payment mechanisms), to security (e.g., how to make sure the goods are delivered to the right person), to performance (e.g., how to efficiently detect illegal copies). We have illustrated some of these problems, and their possible solutions, with examples from the Stanford Digital Library Project.

References

[1] Jose L. Abad-Peiro, N. Asokan, Michael Steiner, and Michael Waidner. Designing a generic payment service. *IBM Systems Journal*, 1998.

[2] AT&T SecureBuy Service. Outsource the technology and save. Available at: `http://www.att.com/easycommerce/securebuy/outsource.html`.

[3] S. Brin, J. Davis, and H. Garcia-Molina. Copy detection mechanisms for digital documents. In *Proceedings of the ACM SIGMOD Annual Conference*, San Francisco, CA, May 1995.

[4] David Chaum. Security without identification. *Comm. of the ACM*, 28(10): 1030–1044, October 1985. Available at: http://www.digicash.com/news/archive/bigbro.html.

[5] CyberCash Inc. Ensuring secure payment for internet commerce. Tech. Report, CyberCash, Inc., 1997. Available at: http://www.cybercash.com/cybercash/wp/merchwp.html.

[6] Dice Corp. Watermarking audio. Available at http://www.digital-watermark.com.

[7] DigiCash. DigiCash brouchure. Available at http://www.digicash.com/publish/digibro.html.

[8] Digimarc Corp. Watermarking images. Available at http://www.digimarc.com.

[9] H. Garcia-Molina, L. Gravano, and N. Shivakumar. dSCAM: Finding document copies across multiple databases. In *Proceedings of International Conference on Parallel and Distributed Information Systems (PDIS'96)*, Miami Beach, Florida, December 1996.

[10] Paul Goldstein. *Copyright's Highway*. Hill and Wang, 1994.

[11] Dan Goodin. A Market Waiting to Happen. *The Recorder*, July 1997.

[12] L. Gravano, H. García-Molina, and A. Tomasic. The effectiveness of *GlOSS* for the text-database discovery problem. In *Proceedings of the 1994 ACM SIGMOD Conference*, May 1994.

[13] Frank Hartung. Links to watermarking sites. Available at http://www-nt.e-technik.uni-erlangen.de/h̃artung/.

[14] iCat Corp. iCat electronic commerce suite 3.0. 1997. Available at: http://www.icat.com/products/suitefacts.htm.

[15] IBM infoMarket Development. IBM cryptolope containers. Tech. report, IBM Corp., 1995.

[16] Steven P. Ketchpel, Hector Garcia-Molina, and Andreas Paepcke. Shopping models: A flexible architecture for information commerce. In *Proceedings of the Fourth Annual Conference on the Theory and Practice of Digital Libraries*, 1997. At http://www-diglib.stanford.edu/cgi-bin/WP/get/SIDL-WP-1996-0052.

[17] Steven P. Ketchpel, Hector Garcia-Molina, Andreas Paepcke, Scott Hassan, and Steve Cousins. U-PAI: A univeral payment application interface. In *The Second USENIX Workshop on Electronic Commerce Proceedings*, 1996.

[18] Bruce T. Krulwich. An agent of change. Available at: http://bf.cstar.ac.com/bf/article1.html.

[19] Microsoft Corp. A foundation for doing business on the internet. Tech. Report, Microsoft Corp., 1997. Available at: http://www.microsoft.com/commerce/whitepaper.htm.

[20] Mondex International. Internet open trading protocol. Tech. Report, Mondex International, 1997.

[21] Netbot Inc. Parallel pull from the invisible web. Available at: http://www.netbot.com/company/tech/tech.html.

[22] Netscape Communications Corp. Netscape merchant server white paper. Tech. Report, Netscape Communications Corp., 1997. Available at: http://www.netscape.com/comprod/products/iapps/capps/mersys_white_paper.html.

[23] Open Market Inc. OM-Transact: A technical overview. Tech. Report, Open Market Inc., December 1996. Available at: http://www.openmarket.com/.

[24] Oracle Corp. Business-to-consumer internet commerce and the role of oracle internet commerce server. Tech. Report, Oracle Corp., May 1997. Available at: http://www.oracle.com/products/web/ics/html/ics_wp.html.

[25] G.J. Popek and C.S. Kline. Encryption and secure computer networks. *ACM Computing Surveys*, 11(4):331–356, December 1979.

[26] P. E. Ross. Cops versus robbers in cyberspace. *Forbes Magazine*, pages 134 – 139, September 9 1996.

[27] Andreas Schöter and Rachel Willmer. Digital money online: A review of some existing technologies. Tech. Report, Inter://trader, February 1997. Available at: http://www.intertrader.com/library/DigitalMoneyOnline/dmo/dmo.htm.

[28] M. Schunter. The SEMPER architecture. Tech. Report, SEMPER, July 1997. Available at: http://www.semper.org/deliver/javadoc.970723/semper.html.

[29] SET Implementors. SET Secure Electronic Transaction 1.0. Tech. Report, Mastercard, May 1997. Available at: http://www.mastercard.com/set/specs2.html.

[30] N. Shivakumar and H. Garcia-Molina. SCAM: A copy detection mechanism for digital documents. In *Proceedings of 2nd International Conference in Theory and Practice of Digital Libraries (DL'95)*, Austin, Texas, June 1995.

[31] N. Shivakumar and H. Garcia-Molina. Building a scalable and accurate copy detection mechanism. In *Proceedings of 1st ACM Conference on Digital Libraries (DL'96)*, Bethesda, Maryland, March 1996.

[32] N. Shivakumar and H. Garcia-Molina. Computing iceberg queries effciently. Tech. report, Stanford Database Group Tech. Report, November 1997.

[33] N. Shivakumar and H Garcia-Molina. Wave-Indices: Indexing evolving databases. In *Proceedings of ACM SIGMOD International Conference on Management of Data (SIGMOD'97)*, Tucson, Arizona, May 1997.

[34] N. Shivakumar, H. Garcia-Molina, and C.S. Chekuri. Filtering with Approximate Predicates. Tech. report, Stanford Database Group Tech. Report, November 1997.

[35] Olin Sibert, David Bernstein, and David Van Wie. Securing the content, not the wire, for information commerce. Tech. Report, InterTrust Technologies Corp., http://www.intertrust.com/architecture/stc.html, 1996.

[36] Marvin Sirbu and J.D. Tygar. NetBill: An Internet commerce system optimized for network delivered services. In *Proceedings of 1995 IEEE CompCon*, 1995.

[37] Mark Stefik. *Letting Loose the Light: Igniting Commerce in Electronic Publication,* In *Internet Dreams: Archteypes, Myths, and Metaphors*, MIT Press, edition, 1996.

[38] L.H. Stein, E.A. Stefferud, N.S. Borenstein, and M.T. Rose. The Green commerce model. Tech. report, First Virtual Holdings Incorporated, October 1994. Available at http://www.fv.com/tech/green-model.html.

[39] Sun Microsystems. Java commerce API user's guide. 1996. http://java.sun.com/products/commerce/docs/ssl/API_users_guide.html.

[40] J. M. Tenenbaum, T. S. Chowdhry, and K. Hughes. eCo system: CommerceNet's architectural framework for internet commerce. Tech. Report, CommerceNet, April 1997. Available at: http://www.commerce.net/eco/eco_2.doc.